TUT 3: GPU Computing (SKM)

Massively parallel processors are replacing standard few-core CPU architectures for high performance computing. Graphics Processing Units (GPUs) feature hundreds of computing cores and require to adopt a parallel programming paradigm from the beginning. This tutorial focuses on how to put GPUs to use for physics and gives an overview over the architecture, programming environments, and basic algorithms.

Time: Sunday 16:00–18:15 Location: H4

Tutorial TUT 3.1 Sun 16:00 H4

High-performance computational physics on graphics processing units — •Tobias Kramer — Universität Regensburg, Germany

Single core computers are not gaining significant speed anymore and the future compute techniques require to program massively parallel processors. Graphics processing units (GPUs) consist of hundreds of processors and have the potential to speed-up computational physics codes tremendously.

To take advantage of GPUs requires to adopt the parallel programming paradigm from the beginning. The GPU tutorial gives first an overview over the GPU architecture, programming environments, basic algorithms. The second part focuses on applications to physical models, including coherent transport, interaction effects, and excitonic energy transfer in light-harvesting complexes.

Source code for some examples is available at
http://quantumdynamics.wordpress.com

15 min. break

Tutorial TUT 3.2 Sun 17:15 H4

Simulating spin models on GPU — •Martin Weigel — Applied Mathematics Research Centre, Coventry University, Coventry, United Kingdom

Over the last couple of years it has been realized that the vast computational power of graphics processing units (GPUs) could be harvested for purposes other than the video game industry. This power, which at least nominally exceeds that of current CPUs by large factors, results from the relative simplicity of the GPU architectures as compared to CPUs, combined with a large number of parallel processing units on a single chip. To benefit from this setup for general computing purposes, the problems at hand need to be prepared in a way to profit from the inherent parallelism and hierarchical structure of memory accesses.

In this tutorial, I will discuss the performance potential for simulating spin models, such as the Ising or Heisenberg models as well as the Edwards-Anderson spin glass, on GPU as compared to conventional simulations on CPU. Different algorithms, including Metropolis and cluster updates, as well as computional tricks such as multi-spin coding are taken into account.